# A Network Based Approach to Data Access, Visualization, Interactive Analysis and Distribution

**Lee Elson, Mark Allen, Jeff Goldsmith, Martin Orton and William Weibel**

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California

*Corresponding author address:* Dr. Lee Elson, M/S 183-510, Jet Propulsion Laboratory, 4800 Oak Grove Dr., Pasadena, CA 91109
E-mail: elson@magus.jpl.nasa.gov

## ABSTRACT

In today's heterogeneous computing environment of proliferating platforms and operating systems, the Internet, through the World Wide Web (WWW), is becoming the preferred interface to much of the world's archive of digital data. WWW browsers are capable of making the computing environment transparent to users, providing easy-to-use access to data archives and applications for studying data. As this process evolves, there will be fewer restrictions as to where data and applications will be required to be stored. By manipulating data at their source, Internet bandwidth requirements can be reduced. We are developing an Internet-based, multi-platform visual computing environment which capitalizes on this trend and provides scientists, working on data either singly or in groups, with the ability to share tools, regardless of the computational platforms being used. We will describe the current state of this freely available package as well as planned future developments.

## 1. Introduction and background

Computers have been used in the analysis of data, both meteorological observations and model output, for well over a generation. The recent explosive growth in both the use of the Internet and in the collection of observational data, together with new requirements for sharing observations and interpretations, have brought new challenges to tool developers and users. This challenge has been intensified by the rapid evolution of computer hardware, software and network technology. Our group has been chartered with creating visualization and analysis interfaces between research scientists and data. During the past decade, our approach has changed from one focused on standalone client software on high-end Unix workstations to one in which multiple platforms, running different operating systems, must communicate over networks. This article will describe our progress, including examples of analysis tasks using our freely available software package, and what we hope to achieve in the future.

There are 2 components to our approach. One involves the transfer of information among computers and among people, and the other involves techniques for understanding data through visualization. It is useful to provide a brief history of how our approach has evolved in both of these areas.

One way of categorizing visualization packages is to distinguish between those which use procedural programming languages such as the Interactive Data Language (IDL), sold by Research Systems Inc[1], and those which use so-called visual programming approaches. The former allows the technically knowledgeable user a great deal of flexibility in the creation of an analysis or visualization package but requires the generation or modification of computer code. The latter can be easier to use "out of the box" but may be less flexible. Adopting the second approach, we developed an application called LinkWinds (Jacobson et al., 1994), which is distributed for free. As its name implies, LinkWinds allows the user to interactively link data, control and display components in windows on the workstation screen, to produce a unique environment

[1] http://www.rsinc.com

suitable for the task at hand. LinkWinds is designed to be intuitive to use, in part because it requires no knowledge of programming by the user. Many users can learn its capabilities by trial and error, referring occasionally to its integrated help system. By writing LinkWinds' code in the C programming language and using X-Windows for the graphical user interface (GUI), graphical display has been optimized for speed and efficiency. This was crucial since at the time that LinkWinds was conceived, visualization requirements pushed the limits of workstation hardware, especially since we sought to provide the capability to handle large data sets.

The emergence of the World Wide Web (WWW) as a tool for distributing data suggested to us that LinkWinds could be quite useful as a browser 'helper application'. Just as a simple image viewer allows the interpretation of gif[2] files downloaded from the WWW, LinkWinds allows actual data files in certain formats to be interpreted and analyzed directly, after being downloaded from a Website. Working closely with users, we determined that the transfer of information between people is as valuable as information transfer between computers when carrying out research tasks. LinkWinds addresses this need through a utility called Lynx that allows the duplication of an analysis session on another machine running LinkWinds and connected to the Internet. Lynx is based on an internal journaling capability that also allows previous sessions to be 'replayed'.

[2] Graphical Interchange Format

## 2. New requirements, new approaches

In 1997, it became obvious to us that we needed a new approach to creating a package for visualization and analysis. Several factors contributed to this conclusion. Perhaps most importantly, the rapid growth in desktop computer processor speed and memory size, together with an increase in the use of specialized graphics hardware and a decrease in prices, brought about a shift in the number and type of desktop computers in use in the scientific community. Many users have discovered that Personal Computers (PC's) running either Microsoft Windows or Apple's Macintosh Operating System (OS) provide a cost effective platform for research related activities including browsing the WWW and reading email. These users are generally not skilled at running applications under Unix operating systems. We rejected the idea of porting LinkWinds to PC's since we would have had to devote significant resources to the creation and maintenance of such ports at the expense of developing new functionality.

A second factor that influenced our conclusion was the growing popularity of the WWW for distributing data over the Internet. Not only are there large, rich data sets available online, projects such as NASA's Earth Observing System (EOS) promise a dramatic increase in the amount of available data in the next few years. As we discuss below, large data sets present both an opportunity and a challenge when one considers the limited bandwidth of the Internet and the limited capabilities of some desktop computers.

There are 2 additional trends in the analysis of scientific data, made possible by the WWW, which were apparent to us. There is a movement towards what is generally called 'outreach' activity. Outreach efforts require that interpretations of data be made understandable to the non-specialist. Second, geographically separated researchers are tending to collaborate more than in the past. Email, together with the ability of the WWW to find and exchange documents and data has fueled this trend.

Given these observations, we decided to create a freely available visualization and analysis package whose operational components can be distributed over the Internet through its object-oriented[3] design, that inherits the performance and functionality of LinkWinds, and a package that functions independently of computer platform and operating system. We felt that Java (Campione and Walrath, 1998; Java Quick Guide, 1999), developed originally by Sun Microsystems, was most likely to fill these requirements. In making the decision to develop a package using Java, several issues that are important to developers and users need to be addressed.

## a. Performance

Java has evolved with the WWW in mind. It has the ability to transfer an executable program, called an Applet, from a WWW host to a client's browser. A frequent observation of this type of WWW application is that it is slow. There are several reasons for this. First, before an Applet can produce results, it must be transferred to the client's machine. Once there, the browser must load and execute the Applet. Various browsers do this with differing efficiency since they have different standards for interpreting Java.

[3] A system designed as a collection of cooperating program elements that consist, in part, of variables with unique values.

These differences can also result in execution errors. Applets also have limits to what they can do. For example, in many cases, security considerations prevent Applets from writing to a client's disk.

Applets were designed to fit into today's WWW culture where users fill out forms on a Web page and await a fairly simple result. In some ways, this is not a very efficient way of using the Internet. Java addresses this through the use of Applications, which do not need a WWW browser. Applications allow performance to be enhanced in several ways. If an Application is to be used many times, it can be downloaded once, saving network bandwidth for other tasks like the transfer of data. Also, although Java is an interpreted language and therefore executes slower than a compiled language, Just-In-Time (JIT) compilers have been developed which significantly increase execution speed. We have found that through careful programming, we can achieve performance that is very good for most[4] desktop platforms used today.

## b. Availability and standardization

A second major design goal for Java was to enable the same code (called bytecode) to be interpreted on any Java enabled platform. This action is carried out by an interpreter, called a Java Virtual Machine (JVM). As noted above, uniformity is difficult to achieve unless developers of operating systems and applications adopt standards. Although Sun Microsystems has kept Java an open (i.e. non-propriatary) system, not all organizations involved with Java have agreed on a set of standards. At the present time

[4] We recommend a processor speed of at least 100 MHz and at least 16 MB of memory.

however, many manufacturers[5] have implemented JVM's which adhere to standards well enough to allow us to produce a working prototype visualization package. Sun has developed a JVM that runs under Microsoft Windows (95/98/NT) as well as one that runs under it's own Unix operating systems. Perhaps most important, these manufacturers have made their JVM's available for free. This is crucial for us since there would not be much point to our creating a freely available package if users were required to purchase the underlying interpreter. Although future actions with regard to standards are unpredictable, we are optimistic that there is a critical mass of cooperating manufacturers committed to the continued viability of Java.

### b. Security and distributed computing

Security and functionality are often inversely related. Any WWW (or Internet) based visualization package will need to be able to at least write out data to the local disk. As we shall see, the full potential of a Java based package will require the ability to transfer data and computational instructions from one platform to another. Security is a primary concern in the design and evolution of Java. For example, the language itself has many safety features that make it difficult to inadvertently (or maliciously) violate safety rules. Java verifies that bytecode has not been altered. Also, read/write/execute access restrictions can be configured easily in the latest version (1.2) of Java. Although these features cannot make an application risk-free, they allow us to build a reasonable amount of security into our visualization package.

[5] Apple, Silicon Graphics, IBM, Hewlett-Packard and others.

As mentioned above, Java was designed for network based distributed computing. The ability to invoke methods[6] from one platform which are resident on another platform or to bundle up a set of instructions and move them to another platform for execution are part of the standard Java package. This very powerful feature will play a significant role in our future development efforts.

## 3. WebWinds: A Java Visualization Package

In designing our new Java visualization package, we felt that an evolutionary approach would work best. Since the development of LinkWinds gave us extensive experience in both what and what not to design into such a package, we began by building much of LinkWinds functionality in Java. In order to highlight both the inheritance from LinkWinds and the new focus on the WWW, we named this new package WebWinds (WebWinds, 1999).

During the initial development of WebWinds, we encountered difficulties due to incomplete implementations of Java on several platforms. This is to be expected when dealing with a new programming environment with ambitious goals, as is the case with Java. For this reason, and because we have extensive experience in building our own GUI's, much of WebWinds bypasses pre-built Java components in favor of our own custom versions. This allows us greater control over both the 'look and feel' and functionality of our package.

In order to maximize the utility of WebWinds as quickly as possible, we concentrated on 4 areas of development. First, we incorporated as much of the display

[6] A procedure, like a subroutine, which performs a certain action.

and control functionality of LinkWinds as we could. Second, we built data ingestion tools and capabilities to make it as easy as possible to bring data into WebWinds. Third, we ensured that collaborative, Internet based capabilities were present and fourth, we produced documentation and built packages that made it as easy as possible to download, install and use WebWinds. The discussion that follows describes the capabilities of our 4th beta version, released in January of 1999.

## a. WebWinds applications suite

Space limitations prohibit a detailed discussion of WebWinds' application tools. Instead, the reader is referred to our online (and downloadable) documentation (Applications, 1999). Here we will summarize these tools and provide examples in the next section. As shown in Table 1, there are 3 types of application tools. Most of the Display tools are able to handle data with up to 3 dimensions (3D). Two dimensions are displayed while a third can often be changed. The displayed image is called a 'slice'. Display Filters allow the user to select which 2 dimensions are viewed. They also allow the user to select a portion of a large data set for viewing. Control applications act to modify Display applications. Some examples are given in the next section.

As was the case with LinkWinds, the fundamental design feature of WebWinds is the ability to connect, or link together both applications and other elements (collectively called 'objects') such as those containing data. Each object occupies a window and windows can be connected using several different buttons. Operations that transfer data are connected through the use of a "drag 'n drop" button. In some cases, sequential operations are possible so that these objects can act first as a 'consumer' of information and later as a 'provider' of information which they may have modified. Control information, such as that provided by sliders, is exported via a "link" button. Examples of both types of connections are given below.

The reason for creating objects in windows and connecting them together is that the user is given flexibility in creating a session. For example, it is common to have one Slider control 2 different Image displays. Other types of controls (not listed in Table 1) are unique to a display, and are therefore enabled via a menu button on that display. For example, selecting 'Crosshair', 'Bounding Box' or 'Grid' is only relevant to the display in which it is selected.

## b. Data ingestion and output

In order for WebWinds to be useful, it must allow data to be brought into applications easily. This required us to build tools to read data files in a variety of formats. For simplicity, we have categorized files into 2 classes: self-describing and raw. Self-describing file formats, such as Network Common Data Format (NetCDF, 1999), Hierarchical Data Format (HDF, 1999) and HDF-EOS (HDFEos, 1999) contain metadata[7] as well as data and usually can be identified by a special 'magic number' at the beginning of the file. Raw data, on the other hand, either contain no metadata or contain metadata that we do not read. Examples of this type of format include files containing only binary (e.g. floating point) numbers or 'home grown' files where the creator has added non-standard header information at the beginning. Another common example of a raw file format is ASCII, or textual data.

[7] Data about data.

WebWinds' approach to the ingestion of data is to use a Data Wizard, or GUI-based input application, to do as much as possible automatically, providing a default configuration, and then to provide the user with several methods of adding to or modifying that configuration. Self-describing files often provide enough information for an initial analysis of a data file without any user input. Raw files require the user to specify such fundamental parameters as the number and size of data dimensions.

Successful data ingestion can involve more than just being able to read certain file formats. Depending on the capabilities of the user's machine and the size of the data file, it may be necessary to restrict the amount of data brought into computer memory. For example, an image with several thousand pixels on a side cannot be viewed on most computer screens without either reducing the resolution of the image or limiting the viewed area. WebWinds allows both of these options, which we call sub-sampling and subsetting, respectively. There are 3 ways in which this can occur. First, if the image is too large to even fit into memory, (e.g. high resolution, multi-spectral images), the user can either subset or sub-sample the data during the loading process. Second, if the image data will fit into memory, but not on the screen, the Display tools will automatically sub-sample the image so that it fits on the screen. Finally, the user can subset or sub-sample a data object after it has been read into memory using the Decimate/Subset Display Filter.

Data files used by WebWinds may be on the user's local disk or, as long as each file has a URL[8], anywhere on the WWW. As described below, files accessible from a WWW browser can also be imported into WebWinds. This means, for example, that a WWW

[8] Uniform Resource Locator

site that creates data 'on the fly' (e.g. a subsetting interface) can be used to provide input into the package.

Saving results is an important capability for an analysis package. Currently, a data object can be saved in several formats, including NetCDF and raw binary. In section 5, we discuss future enhancements in this area.

## c. Download, installation and documentation

WebWinds' software and documentation is freely available from our WWW page (WebWinds, 1999), once the user has filled out a standard non-commercial license agreement. Although the bytecode is identical for each type of platform, we have created individual packages for Unix/Linux, Windows (95/98/NT) and the Mac in order to facilitate installation. Packages for other platforms (e.g. OS/2) will be created soon. Except for Unix/Linux, each package contains the JVM, WebWinds software, documentation and sample data and is about 20 Mb in size. The Unix/Linux package requires that the JVM (in the form of the Java Development Kit) be downloaded separately from the workstation manufacturer's (or the Linux) WWW site. Unpacking and setup are simple and straightforward for each package if one follows the instructions in the documentation. The amount of disk space required depends mostly on the amount of data that the user would like to store since the software and documentation require little storage media.

If the user wishes to use WebWinds as a browser helper application, it is necessary to configure the browser for this capability, just as it is for most helper applications. The instructions for doing this vary with the platform and are included in the documentation. If both the browser and WWW server[9] are properly configured, clicking on a link will cause the associated file to be imported into WebWinds.

WebWinds' documentation is written in HTML[10] and is available both on the WWW site and as part of the downloadable packages. Each Application also has a "?" button which, when pressed, brings up a Java browser showing the documentation for that object. Thus, for example, if a user brings up a Histogram Application and doesn't know how to use it, pressing the "?" button will provide a description of that Application. Included in the documentation package are several step-by-step examples of how to set up sessions. These sample sessions either use data included in the package or data available on the WWW. The descriptions include screen shots of the objects being described.

### d. Automatic scripting: Rerun and remote sessions

By default, each WebWinds session generates a set of commands, called a script, which reflect the actions performed by the user. These commands are automatically saved in a file that must be renamed if a permanent version is desired. Any of these script files may be run during a WebWinds session and, with certain restrictions, may be run on other machines. Not only can a user easily re-create a session that was found to be useful, but such sessions can be given to collaborators for their own use. A data provider can set up a WWW site containing both data files and script files. Links to script files on a

[9] The WWW server must be configured to assign the correct mime type to a file. For example, an HDF file should have the type application/x-hdf. The user has little control over the server's configuration.
[10] HyperText Markup Language, the language used by the WWW.

WWW site allow the user to import the scripts into WebWinds so that the process of setting up a session can be made automatic, once WebWinds has been installed. In addition, our development team can use script files to either help users with problems or investigate bugs in the WebWinds package.

The scripting language developed for WebWinds serves another purpose. It enables users to establish remote, shared sessions with other WebWinds users on the Internet. To do this, one user requests a shared session with another party by supplying an Internet address. If WebWinds is active at this address, a dialog box will open there asking for a connection. If the second user approves, all objects opened on the first user's desktop will open on the second user's desktop as well. The second user may also request a shared session with the first user, resulting in a collaboratory environment. In order for shared sessions to work properly, identical data files must be available to both users, either on their respective local disks or on the Internet. Because only scripting commands are exchanged, these collaborative sessions can be carried out over low bandwidth connections.

## 4. Sample Sessions

One way to illustrate the capabilities of WebWinds is to show examples of how it might be used. Because of space limitations, we can only present a very limited introduction to the use of the Applications shown in Table 1. Our WWW site has several additional examples that give a more complete description of how to use the

Applications. Some of the screen shots shown below may differ from what the user will see due to changes in the released version of the software after the time of this writing.

Here, we offer 2 examples of how data might be brought into WebWinds and examined. Each example has a script file associated with it so that the user can create each session by simply selecting the appropriate script. For the first example, the script is included with the distribution package and can be selected under the 'Rerun' option of the 'Utilities' menu. The script describing the second example is available at our WWW site.

## a. Total Ozone Mapping Spectrometer monthly averages

The first example examines data from the Total Ozone Mapping Spectrometer instrument. These data represent monthly averages of total column ozone for 12 months in 1992 and 1993 and are contained in an HDF file. There are 2 approaches that can be used to import these data. One approach would be to use the interactive Data Wizard to select the file and change or add to its metadata. This is the approach used in the second example, in the next section.

Another approach is to insert information about the observations into the WebWinds file, datamanager.txt. This file characterizes data sets used by the package and is organized into Data Folders, Data Sources, file Formats and conversion types. Folders can be nested (one inside the other) so that they can contain other Folders or Data Sources in analogy with (but not directly related to) folders and files on a computer disk. Data Sources specify more than just the location of a source file on a disk. In addition to file name, path and type, they contain information about the dependent and independent variables and associated color palettes. Table 2, modified slightly from the datamanager.txt file distributed with the package, contains a Data Source listing that is used in this example. The first line assigns the name '92-93' to this Source which will appear in the Desktop menu under 'File' -> 'Open'. The second line specifies the name of the file to be used. This can be a URL if appropriate. The third line indicates that the file is in HDF format. The fourth line is not really necessary but is included as a reminder that several simple mathematical conversions can be carried out as data from a file are loaded into the data object. The next 4 lines define the dependent and independent variables, giving them names, axis numbers, units and value ranges. Note that we have changed the units in Table 2 to better reflect the associated variables and that the independent variable, here called 'Ozone' is always called 'axis 0'. The last line assigns a pre-constructed color table, called 'rgb', to the data set.

Since the Data Source just described is similar to that included in the WebWinds distribution, no additions to the distributed datamanager.txt are required, although one might change the units for consistency. WebWinds, when started, first produces a Desktop window as shown in Fig 1a. As described above, from the 'File' menu, first select 'Open', then scroll down to and select '92-93'. This brings up a data object that displays metadata read from the file and from datamanager.txt. Pressing the 'Load' button causes the data in the file to be read and produces the data object represented in Fig 1b. Next we wish to display these data. We do this by first selecting 'Image' and then 'Slider' from the 'Tools' menu on the Desktop. With the mouse cursor over the "drag 'n drop" button in the upper right corner of the data object, press a mouse button and move the cursor to the Image before releasing the button. Do the same with the Slider. Finally, in order to tell the Slider what it will control, connect the link button (the button with interlocking rings) on the slider to the Image. The results are displayed in Figs 1c and 1d. The Image shows the distribution of column ozone averaged over a particular month. The month displayed can be changed using the Slider. Notice that the Image and Slider both show, in numerical form, the month being displayed in the Image.

There are several additional things to note about the Image. One is the presence of a Crosshair that can be moved using the mouse. The value of Ozone under the Crosshair is displayed just above the image itself. Another is that if one selects 'coast180' from the 'Overlays' menu, a white outline of the coastlines of the world appears in the Image. Although the script file for this example stops with the displays just discussed, there are several additional features that can be noted. Pressing the 'LinePlot' button on the Image produces an X-Y plot for the point under the Crosshair, as shown in Fig 2a. For this example, this is a plot of column ozone as a function of time at the latitude and longitude of the Crosshair. Notice a significant annual component to the variation in the LinePlot window. As the Crosshair is moved, the LinePlot is updated. A similar tool, called Profile can be selected from the Tools menu. This tool displays an X-Y plot of data across an arbitrary path in the Image. First, use the "drag 'n drop" button to connect the data object to this new tool. Next, follow the directions in the Profile window and link the Image to the Profile. In order to draw a profile on the Image, select 'Draw Profile Line' from the 'Profile Line' option on the Image 'Menu' button. Then use the mouse to actually draw the line on the image. The result, shown in Fig 2b, was produced by drawing a vertical line at 0 degrees longitude. Therefore, it shows ozone as a function of latitude at that longitude. The mid-latitude ozone peaks, as well as the equatorial minimum is clearly visible.

## b. Climatology Interdisciplinary Data Collection

The second example uses data from the Goddard Distributed Active Archive Center (DAAC)[11]. Our intent is to demonstrate that any file accessible on the WWW can be treated as if it were a local file, apart from the latency inherent in the process of downloading the data. We believe that this has relevance to data producers, providers and users.

We have chosen to use products from the Climatology Interdisciplinary Data Collection[12] because of the availability of data representing observations of several atmospheric and surface parameters which have been mapped to a common grid. It is important to note that data need not be on a common grid to be compared in WebWinds. This example differs from the previous one in that the data are not self-describing, i.e. the files contain no metadata. Instead, each file holds data in floating point format and the user must read the documentation to know how to interpret the numbers in the file.

To begin this session, we start WebWinds as in the previous example, obtaining the Desktop window (Fig. 1a). Next we select 'New Source' from the Files menu, obtaining a window similar to that in Fig. 3a. Here, we must enter the URL[13] for the first data file we wish to use in the text pad marked 'Current File'. We've selected surface temperature data derived from the TOVS (TIROS Operational Vertical Sounder) instrument. With the file selected, press the 'Open' to get an ObjectDisplay' window. Because this file is not recognized, the user must select a file type from the 'types' menu button. The appropriate

[11] http://daac.gsfc.nasa.gov/DAAC_DOCS/gdaac_home.html
[12] http://daac.gsfc.nasa.gov/CAMPAIGN_DOCS/FTP_SITE/inter_disc.html
[13] ftp://daac.gsfc.nasa.gov/data/inter_disc/tovs_atmo_sound/tsurf/1988/tovsng.tsurf.1pmegg.8801. bin

value here is 'RawFloat'. When this selection is made, several other required fields appear in the window, as shown in Fig. 3b. Since the data are 2 dimensional, the number '2' must be entered in the 'Number of Dimensions' text area and '360 180', indicating that the data represent a 360x180 (longitude x latitude) array, must be entered in the 'Size of Dimensions' area. Finally, a title should be added to the 'Title' area. We've chosen 'Surface Temperature'. With these specifications complete, pressing the 'Open' button checks to see whether the user has entered illegal choices. If not, another press of the open button causes the window to display a summary of the data attributes, including default metadata (in the 'Meta Data' window) describing the dependent and independent variables. At this point, no data have been read from the file. Pressing the 'Load' button does this, providing information about the data values in the file, as shown in Fig. 3c. Notice that the metadata window shows default names for the variables. In order to change these names, as well as the units and data ranges, the user must click on each of the metadata lines in turn. For example, clicking on the first one, titled 'Value' brings up a separate window. Selecting 'Edit' from the menu button in this new window allows the independent variable to be given the title 'Temperature', as illustrated in Fig. 4a. Notice that the 'Start Value' was changed from -999. to 0. This was done to eliminate the 'no data' values in the file. In order to make this effective, the 'Limit Data' Control Flag must be checked. To make these metadata changes effective, 'Apply' must be selected from the menu. This window can then be closed. The 2 dependent variables, longitude (-180 to 180) and latitude (-90 to 90) (called 'Row' and 'Column' in Fig 3c) can also be edited using the same approach. When all metadata have been modified, the 'Load' button on the object display must be pressed.

With one data object created, it is time to choose a second for comparison. We've selected outgoing longwave radiation (OLR) data derived from the Earth Radiation Budget Experiment (ERBE). Again, starting with the 'New Source' option from the Files menu, we can create an OLR data object. The URL[14] will be different of course, as will the dependent data variable as specified in the metadata window. Be sure to press 'Load' on this new data object, once all modifications have been made to metadata.

With 2 data objects created, we return to the Desktop and select 'Image' from the Tools menu. Use the "drag 'n drop" button in the upper right corner of the Surface Temperature Object Display to connect the Temperature to the Image. Next, select '2D Scatterplot' from the same Tools menu and first drag the Temperature and then the OLR objects into it. As shown in Fig. 5a, the resulting display shows a scatterplot, as well as some simple statistics for the 2 data sets. There appear to be 2 distinct regimes for correlations between OLR and Surface Temperature. Notice that the scatter diagram is color coded. The colors correspond to different locations on the earth's surface. A more precise way to determine the geophysical location of points in the Scatterplot is to select 'Resize Bounding Box' from the 'Bounding Box' submenu of the 'Menu' button on the Scatterplot. As shown in Fig. 5a, this allows a box to be drawn over a portion of the data in the scatter diagram. If we now use the link button (top right button) in the Scatterplot to connect this display to the Image, we see the data points inside the box are highlighted in yellow in the Image window (Fig 5b).

WebWinds has a second tool that is useful for comparing 2 data sets. From the Tools menu, select 'WindowTool'. As with the Scatterplot, first drag the Surface

[14] ftp://daac.gsfc.nasa.gov/data/inter_disc/radiation_clouds/erbe_rad/1988/erbe.lwolr.1nmegg.8801.bin

Temperature Object Display, then the OLR Object Display into this tool. The result, shown in Fig 4b, is a display with Temperature in the background and OLR in a small inset window. The inset can be resized and moved over the background image.

## 5. Future Work

In the next year, we plan to enhance existing features and add new ones. Since saving results is important, we plan to add several file formats to our file save menu, including HDF. We also plan to add a capability to save image results as JPEG[15] or Postscript files suitable for publication. Also on the drawing board are several 3D displays (e.g. data on a globe or a transparent 3D image) and an animation tool that will allow the user to record displays as they change in response to control changes.

As we discussed in Section 2, components of a Java application can be easily distributed over the Internet. This will allow us, for example, to move parts of WebWinds to a server that can access data sets through local, high bandwidth connections. By doing this, the process of selecting only the data that is needed can be made much more efficient. Suppose data are organized at an archive so that very high spatial resolution global, atmospheric observations are stored in large files, each of which represent data taken during a 24 hour period. If a user is interested in a time series of observations over a limited geographical region, the normal process would be to obtain large amounts of data, most of which would be discarded. By placing subsetting software near the data source, the user would be able to avoid this inefficient transfer method, and would be

[15] Joint Photographic Experts Group

able to access the data of interest much more quickly. We plan to develop server packages that would carry out these processes in the near future. We also plan to build interfaces to other database management systems, making it possible to carry out more sophisticated data queries and have the results fed directly into display and control applications.

Further in the future, it should be possible to move parts of WebWinds from one platform to another. Suppose there is a mirror site for a particular data archive. A 'roving' version of WebWinds might assess the load level at several sites to determine which one would perform a function fastest, then move parts of a session to that site. Similarly, a user might enable an auto-update feature that would allow software updates or additions for his client machine to occur automatically, but only when needed. Finally, we intend to create 'builder applications' that will enable novice users to create their own Applications.

The amount of compute power connected to the Internet is enormous. Java offers a method for harnessing that power in a way that is invisible to the user, but highly beneficial to the scientific community.

*Acknowledgments.* This research was performed at the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration.

## References

Applications, cited 1999: WebWinds Applications Suite. [Available online at http://webwinds.jpl.nasa.gov/beta4/webpage/webhelp.html] OK cnc

Campione, M. and K. Walrath, 1998: *The Java Tutorial,* Addison-Wesley, 964pp. [Available online at http://java.sun.com/docs/books/tutorial/index.html]

HDFEos, cited 1999: HDF-EOS.[Available online at http://ecsinfo.hitc.com/iteams/HDF-EOS/HDF-EOS.html]

HDF, cited 1999: Hierarchical Data Format. [Available online at http://hdf.ncsa.uiuc.edu]

Jacobson, A.S, A. L. Berkin and M. N. Orton, 1994: LinkWinds: Interactive Scientific Data Analysis and Visualization. *Commun. ACM.*, 34, 43-52.

Java Quick Guide, cited 1999: A Quick Guide to the Java Platform. [Available online at http://java.sun.com/nav/whatis/index.html]

NetCDF, cited 1999: Network Common Data Format. [Available online at http://www.unidata.ucar.edu/packages/netcdf/]

WebWinds, cited 1999: WebWinds [Documentation and software available online at http://webwinds.jpl.nasa.gov]

## Tables

**Table 1. WebWinds Applications Suite**

| Name | Function |
|---|---|
| Displays | |
| Average | Averages slices of data sets and displays the results |
| Combine | Algebraically combines data sets and displays the results |
| Compare | Computes and displays vector statistics for a 3D data set |
| Contour | Draws contours on several display applications |
| FFT | Displays the results of Fast Fourier Transforms |
| Histogram | Displays data distributions |
| Image | Displays data as 2-D image |
| Line Plot | X-Y plot of data parallel to an axis |
| Profile | X-Y plot of data along a specified line segment |
| Track Pixel | Prints numerical values and statistics for a point or area |
| Value View | Displays numerical values for a slice of data |
| Window Tool | Displays one data set on top of another |
| 2D Scatterplot | Scatters one data set against another (with statistics) |
| | |
| Display Filters | |
| Decimate/Subset | Creates a new data set from a portion of an old one |
| View Axis | Changes the viewing axes of a data set |
| | |
| Controls | |
| Calculator | Allows algebraic manipulation of data for Combine |
| Color Tool | Choose/manipulate color palettes |
| Combine Slider | Choose offsets for Combine only |
| 3-Slider | Choose offsets for Window Tool only |
| RGB Slider | Choose 3 data slices as inputs to a color display |
| Slider | Pick a slice for any application to use |

**Table 2. Excerpt from datamanager.txt**

```
DataSource "92-93"
    file "Ozone_Dec22_090631-0.hdf"
    format "HDF"
    converter ""
    meta "latitude" "degrees" axis 2 entries 180 range 90.0 -90.0
    meta "longitude" "degrees" axis 1 entries 360 range 359.0 0.0
    meta "time" "months" axis 3 entries 12 range 1.0 12.0
    meta "Ozone" "DU" axis 0 entries 254
    colorhint "rgb"
```

**Figures**

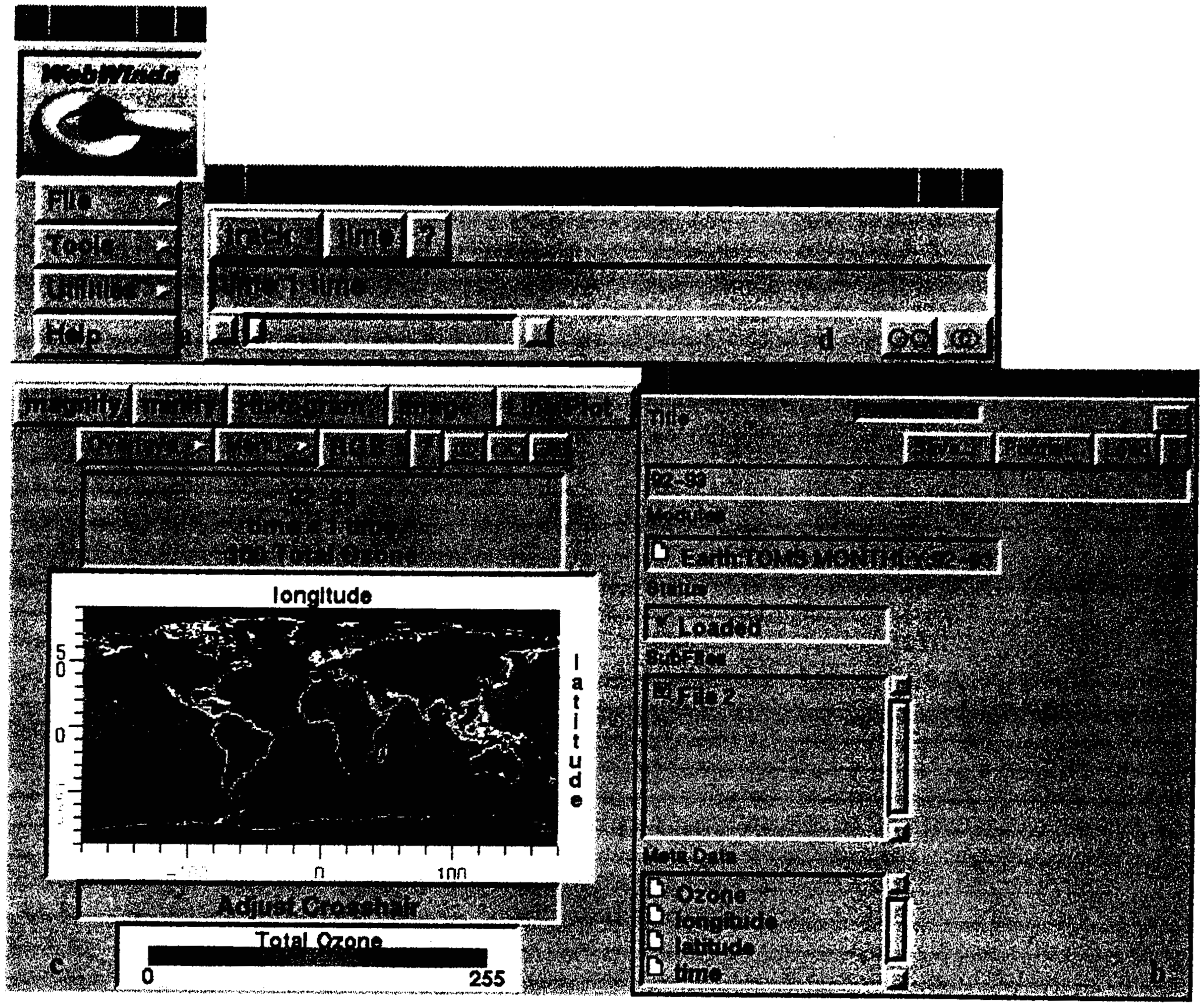


Fig. 1 a) The Desktop is the top level window. b) The Object display for TOMS Ozone c) The TOMS data in an Image display and d) A Slider used to control the time slice.

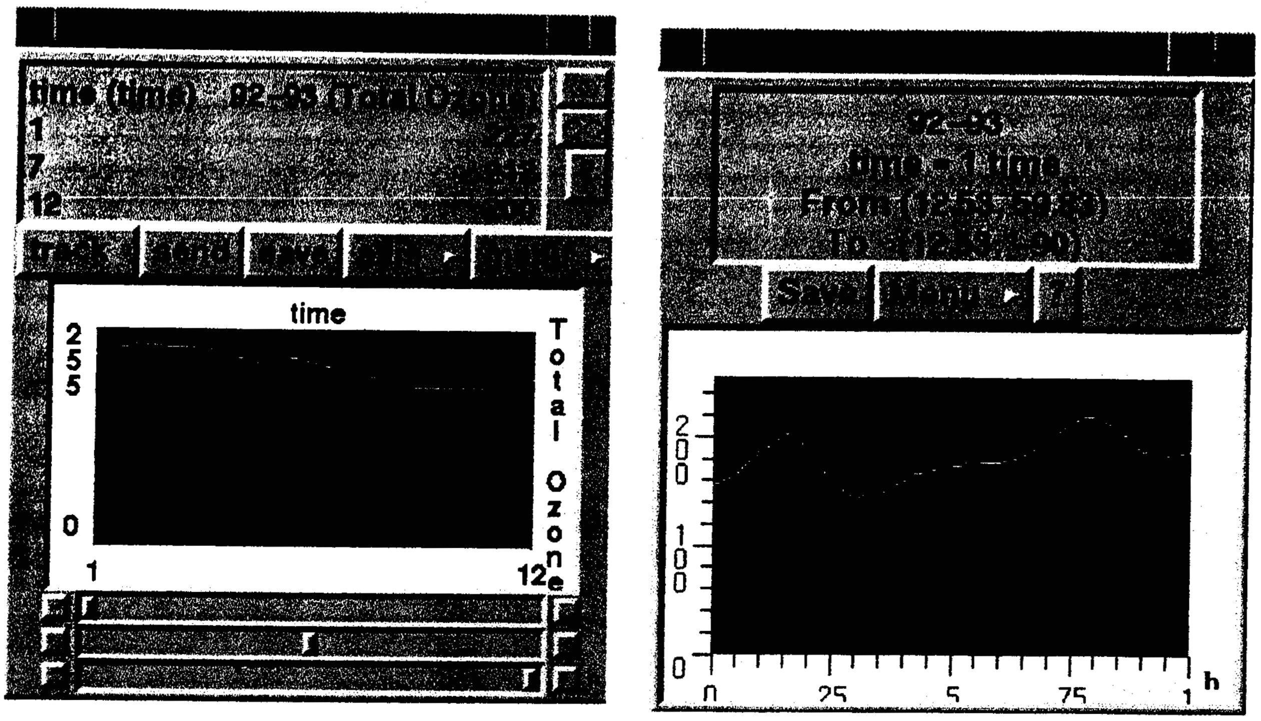
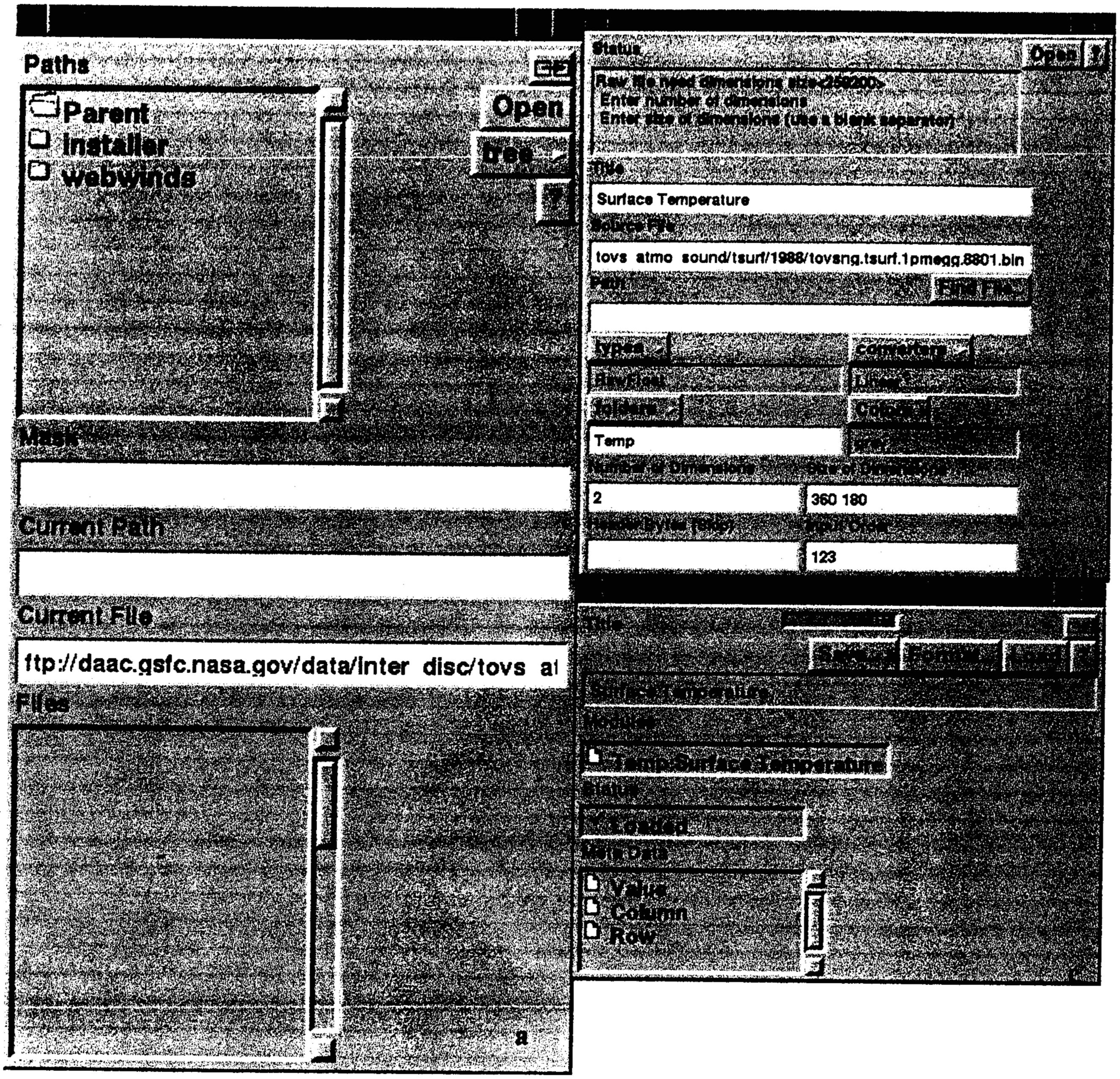


Fig. 2 a) A LinePlot showing Ozone versus time. b) A Profile showing ozone versus horizontal positio



Current File: ftp://daac.gsfc.nasa.gov/data/inter disc/tovs at

Status:
Raw file need dimensions size<259200>
Enter number of dimensions
Enter size of dimensions (use a blank separator)

Title: Surface Temperature

Source File: tovs atmo sound/tsurf/1988/tovsng.tsurf.1pmegg.8801.bin

types: RawFloat

converters: Linear

folders: Temp

Number of Dimensions: 2

Size of Dimensions: 360 180

Input Order: 123

Surface Temperature

Modules: Temp:Surface Temperature

Status: Loaded

Fig. 3. The object display used to a) locate a file, b) specify file parameters, c) represent a loaded data object.

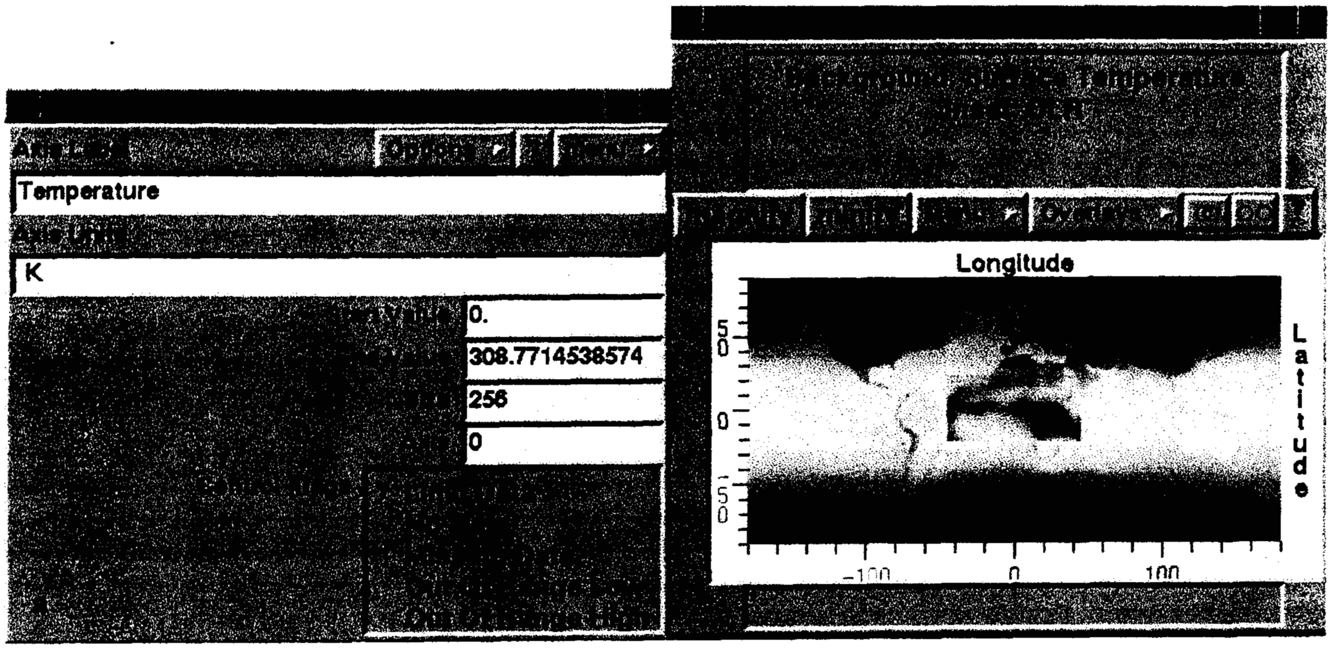
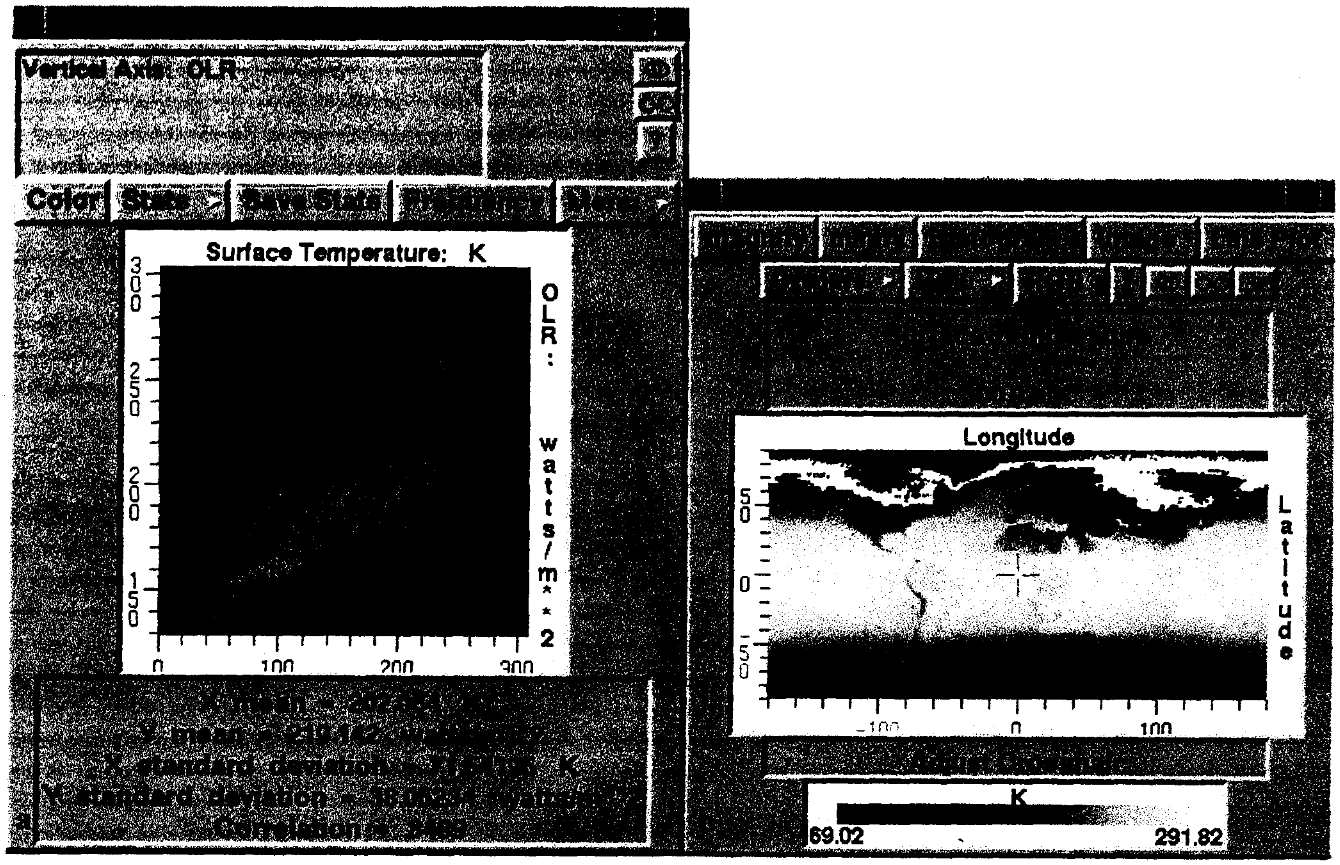


Fig. 4 a) An example of a Metadata object showing labels, units and values. b) A WindowTool object containing an image showing surface temperature in the background and OLR in the small inset windo



```
X mean = [illegible]
Y mean = 210.142 [illegible]
X standard deviation = [illegible] K
Y standard deviation = [illegible]
Correlation = [illegible]
```

Fig. 5 a) a 2-D Scatterplot showing the relationship between surface temperature and OLR. The scatter points are color coded, representing different spatial locations. b) an Image showing the surface temperature data. The yellow area shows the location of the data inside the red bounding box in the 2-D Scatterplot.